

Computing the Data Semantics of WSDL Specifications via Gradient Boosting

Alexandros G. Valarakos¹ and George A. Vouros¹

Abstract. This paper proposes a method for the semi-automatic semantic annotation of WSDL specifications, given ontologies related to the domain of services. The proposed method uses a synthesis of mapping methods to map input/output messages' parameters to ontology classes. Exploiting validated results provided by humans, the method learns via the gradient boosting learning algorithm to combine the individual mapping methods towards improving its accuracy. The aim is to mitigate difficulties concerning mappings and address limitations of other approaches, even in challenging cases, so as to assist human annotators to perform their work. The paper presents experimental results of the proposed methods.

1. INTRODUCTION

The lack of explicit semantics in WSDL (Web Services Description Language) specifications makes them insufficient to satisfy the requirements for flexible and effective web service 'manipulation' tasks, as they force the relevant mechanisms to be based on keyword matches. While we may relate different types of semantics to web services, we can distinguish between (a) Data semantics (introducing the semantic signature of services: semantics of input/output messages of service operations), (b) Functional Semantics (function of operations and of the service itself). This paper focuses on data semantics, which is generally accepted to be one of the most critical aspects regarding services' semantic description. Although other important aspects exist with respect to web services' signature (e.g. goals, pre/post conditions [8], services' classification [2]), they strongly depend on data semantics.

This paper proposes a method for semi-automatic semantic annotation of WSDL specifications, given ontologies related to the domain of services. Towards this objective, we propose a method that (a) maps input/output messages' parameters to ontology classes so as to assist human semantic annotators to unveil the data semantics of services, and (b) learns how to produce accurate mappings based on the feedback it receives from the human annotators. Actually, the proposed method learns how to effectively synthesize a set of mapping methods, with respect to the idiosyncrasies of domains, services and ontologies involved, so as to mitigate difficulties in computing mappings, and limitations of other approaches, even in challenging cases.

We study the performance of the proposed method to a subset of service specifications from the SAWSDL-TC test collection [5]. The SAWSDL-TC test collection is a good candidate for evaluating our method since, beyond the fact that it is a collection written in a standard language for services' semantic annotation (SAWSDL), it provides mappings (WSDL parameter-names to OWL classes) that can be used as gold-mappings.

As pointed out in [9], the process of semantic annotation of services requires input from multiple sources, including the source code of the service, the API documentation and description, as well as "external" information sources such as users' license agreements and sources of background knowledge [2]. The main assumption behind this article is that these information sources, in conjunction

to service specification elements, can provide valuable information for computing the data semantics of web services. It is within the goals of this work to provide assistance to human semantic annotators towards specifying the data semantics of WSDL specifications and reduce human intervention to the tasks that humans can do better: Introduce documentation for WSDL elements and validate results. Specifically, according to the three-stage methodology shown in Fig.1, humans are involved at the early stage of annotating WSDL specifications by introducing natural-language documentation of WSDL elements, and at the validation stage, where they validate the semantic specifications computed. It should be noted that in the following by "annotation" we denote the introduction of the textual annotations and by "semantic annotation" we denote the annotation using ontology elements. This overall task takes as input a WSDL document and related domain ontologies, and produces a semantic description of the corresponding web service's profile by exploiting WSDL specifications and textual descriptions of services' elements.

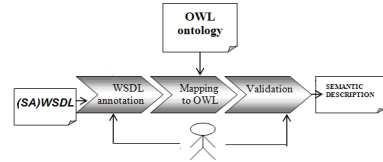


Figure 1. The WSDL semantic annotation process

The paper is structured as follows: Section 2 states the problem that this article deals with. Section 3 presents related work, motivates the proposed method and states its contributions. Section 4 presents in details the proposed semantic annotation system. Section 5 presents and discusses experimental results and section 6 concludes the paper, highlighting important aspects and sketching future work.

2. PROBLEM STATEMENT

Considering services' signatures specified in WSDL we deal with the following specifications:

- The signature of a service s specifies a set of input/output messages. These are denoted $\langle \text{service_id}, \text{name}, \text{message_type}, \text{message_id} \rangle$, where message_type can be *input* or *output* and "name" is the name of the message. The service is related via its WSDL *documentation* element with *annotation_text* that aims to describe the service as a whole (e.g. the functionality of a service – although not restricted to this).

- Each message $\langle \text{service_id}, \text{name}, \text{message_type}, \text{message_id} \rangle$ has one or more parameters: $\langle \text{service_id}, \text{message_id}, \text{name}, \text{data_type} \rangle$, where name is the name of the parameter, and data_type specifies the (atomic or complex) data type of the parameter. Again, each message is related via its WSDL *documentation* element with an *annotation_text* that aims to describe the communicative intention of the message providing the appropriate context to disambiguating message parameters.

- Each parameter is associated with textual annotations: $\langle \text{service_id}, \text{message_id}, \text{name}, \text{annotation_text} \rangle$, where the *annotation_text* aims to describe the intended meaning of the parameter.

¹ AI Lab, Information and Communication Systems Eng., Dept University of the Aegean, Samos, 83 200, Greece {alexv, georgev}@aegean.gr

The main problem that this paper deals with is as follows: “Given (a) a WSDL specification and (b) an ontology $O=(S,A)$, S being the ontology signature and A the ontology axioms, provide mappings of WSDL messages’ parameters to ontology classes with respect to the intended semantics of the service”. Computed mappings between WSDL messages’ parameters and ontology elements are of the form “<service_id, message_id, name, element, rel>” where *name* is the name of the parameter, *element* is a term in the ontology signature S , and *rel* is the assessed relation between *element* and *name*: Considering also *name* to denote a set of individuals, this relation may be equivalence, or subsumption. For the purposes of this paper we restrict our attention to equivalences.

During the validation stage, for each service parameter, human annotators have to select one of the top-k ranked mappings provided, or specify a new one when no correct mapping exists in the k proposed ranked mappings. Variable “k” is configured by the annotator and sets the maximum number of the proposed ranked mappings.

To compute services’ data semantics automatically, by exploiting their WSDL specifications, is a very ambitious goal. This is so since, pitfalls related to phenomena concerning synonym terms, homonym terms (i.e. polysemy), typographic variations of terms, differences between the granularity of services’ and ontological specifications, have to be mitigated. In conjunction to these pitfalls, we need to take also into account the “nature” of WSDL specifications, which are being produced from program code, with few and, in most of the cases, misleading comments, descriptions and “tricky” names of the parameters being involved, as well as with improper or faulty use of domain terminology. To address these issues we are targeting a mapping mechanism that (a) integrates different methods for computing mappings of WSDL messages’ parameters to ontology classes, and (b) it can exploit known mappings so as to learn how to combine these individual mapping methods, increasing its accuracy. Eventually, the mapping mechanism will be precise enough to suggest the correct mappings, providing great help to human annotators.

3. RELATED WORK & MOTIVATION FOR RESEARCH

Close to the aims of our work are efforts that exploit textual descriptions of services for the annotation, classification, and for the assessment of similarities between web-services.

The METEOR-S Annotation Framework [1] is one of the most prominent approaches, aiming to marking up web service descriptions with ontologies semi-automatically, by means of a schema matching algorithm: Ontologies and XML schemata used by WSDL specifications are converted to SchemaGraphs that are compared by computing linguistic similarities between elements and structural similarities between the schemata. For the purposes of linguistic similarity, the proposed algorithm consults WordNet to find synonyms. In this paper we emphasize on the combination of various methods for the exploitation of different facets of available information, towards “uncovering” the semantics of WSDL parameters. In contrast to our approach, METEOR-S exploits “shallow” features concerning XML schema and ontology elements’ names: This affects methods’ efficacy in cases where polysemous terms appear, or in cases where specifications are at different granularity levels, as far as the conceptualization of the domain is concerned.

The work reported in [2] aims at assessing the similarity of WSDL specifications by exploiting the structure of data types and operations of services, as well as the semantics of natural language descriptions and identifiers. Although the aim of this work is to support query-by-example discovery of services, the emphasis on semantic matching given textual descriptions of services, and the exploitation of identifiers’ semantics, brings this work close to our work. This approach uses a vector model, exploiting the textual

descriptions of services, and consults WordNet to calculate the semantic distances between identifiers of WSDL elements. The semantic matching is restricted to the exploitation of identifiers, which are not disambiguated, making the calculation of semantic distances rather problematic. Disambiguation is a vital task in our research since we aim to match WSDL elements to specific ontology concepts, explicating their semantics.

Aiming to show how content-based approaches can contribute to semantic matching of OWL-S service specifications, OWLS-MX [3], aims to exploit the implicit semantics of any part of OWL-S service description by representing it as a weighted category-index term vector. Index terms are stemmed lexical items from a shared minimal vocabulary. This vocabulary results from the canonical unfolding in an underlying ontology of the words or concepts that exist in text categories that correspond to OWL-S elements (hasInput, ServiceName, TextDescription etc). Although the aims of this work are quite different from our aims, it provides firm evidence towards our conjecture: That the use of textual descriptions of service parts in conjunction to their formal specifications can help to uncovering their implicit semantics. However, as shown in [4], pitfalls to the matchmaking methods require logic-based matching to be augmented with precise computations of non-logic-based matches to improve the overall performance of the method.

Further evidence is provided by experiments with the ASSAM’s annotation wizard [7]. ASSAM casts the problem of classifying operations and datatypes in a Web Service as a text classification problem. The tool learns from Web Services with existing semantic annotations. Given this training data, a machine learning algorithm can generalize and predict semantic labels for previously unseen Web Services. The approach proposed in the current paper aims at learning how to improve the precision of combining individual mapping methods given input from human annotators. The method at its very beginning does not require the pre-existence of training examples to decide on the mapping of WSDL elements to ontology concepts.

Concluding the above, the method proposed here combines features from the previous approaches: In close relation to METEOR-S and ASSAM, and in contrast to the other approaches described, we aim at semantically annotating web services’ input/output parameters with ontology elements, by computing direct, exact mappings of these elements to ontology classes. However, in close relation to OWLS-MX but in contrast to METEOR-S and ASSAM we aim at showing that the exploitation of services’ content (in conjunction with textual descriptions) by a combination of mapping methods can provide great help to annotating services. In addition to the above and to a great extent than the described approaches, we propose a method that combines mapping methods (to address different phenomena related to computing mappings), and gradually learns the effective combination of the different mapping methods, given correct mappings being suggested by human annotators during the validation stage. Finally, as a by product of this research we have produced corpora that can be used as benchmarks for evaluating similar methods: We believe that these are important contributions towards advancing the automatic semantic annotation of WSDL specifications.

4. SEMI-AUTOMATIC SEMANTIC ANNOTATION OF WSDL SPECIFICATIONS

4.1 Overall Method

Considering the stages shown in Figure 1, during the annotation stage, humans provide annotation texts for the elements of the WSDL specification.

The mapping stage takes as input a WSDL document, a set of domain-related ontologies and known correct mappings provided

by previous executions of the method. The output is a set of ranked mappings between WSDL messages' parameters and ontology elements provided by a combination of mapping methods. The mapping tool, in addition to the parameters names of the input/output messages, exploits services' types' specifications, as well as textual descriptions-annotations and comments related to WSDL elements.

Finally, in the validation stage, users inspect the assessed mappings to validate the results. Users may change the mappings computed in the previous stage. The validated mappings are fed to the mapping stage during subsequent executions of the method.

4.2 Mapping Methods

Although any set of different methods may have been combined, in this work we have chosen and combined methods that exploit different facets of the information available and address different pitfalls to the mapping of elements: Methods that measure string similarity (COCLU and Levenshtein), a method that computes and exploits latent features of specifications (LSA), and a method that exploits vectors of "surface" features of specifications (VSMM). These methods are briefly described in the next paragraphs. Before proceeding to the description of the individual mapping methods, it is necessary to point out that the compound terms attributed in a "name" are split into single words, considering some special characters (i.e. the "-" and "_") as words' delimiters, as well as the change of characters' cases from lower to capital. For example, the compound word "AcceptCostAndHealingPlan_CostAndHealingPlan" is tokenized into the "Accept", "Cost", "And", "Healing", "Plan", "Cost", "And", "Healing", "Plan" words. The new set of words enriches the textual annotation of the specific parameter to which the compound word corresponds. Furthermore, viewing the WSDL specification as a hierarchical structure rooted by the service itself and including several operations that further comprise several messages, each with its own parameters, we may assume that information that exists in the nodes in this hierarchy is inherited to subsidiary nodes, providing contextual information for further disambiguating the meaning of messages' parameters.

COCLU is a compression-based clustering algorithm (COmpression-based CLUstering). The algorithm is based on the assumption that different lexicalizations of a term (typographic variants) use a common set of "core" characters. Therefore, typographic variants that "mostly" use this set are potential alternative lexicalizations of the same concept. Further details for this model-based and partition-based clustering algorithm are provided in [6]. COCLU is used for the comparison of (a) the value of the "name" attribute of each parameter, with the labels of ontology classes, (b) the value of the "name" attribute of each parameter, with the local name of ontology classes, and (c) the documentation of the parameter with the comments of ontology classes.

In addition to COCLU, we have used the Levenshtein string matcher that incorporates a distance measure. This specifies the minimum number of operations needed to transform one string into another. The valid operations are: insertion, deletion, or substitution of a single character. Our implementation considers a match between two strings if and only if at most two operations are required for their transformation.

The LSA-based-mapping method [10] aims to disambiguate the meaning of WSDL messages' parameters by mapping them to WordNet senses that best capture their intended meaning. The same method maps ontology classes to WordNet senses that capture the human-intended meaning of the formal specifications: Having done these mappings, and in case the intended meanings (WordNet senses) of a class and of a service parameter coincide, then these are mapped. The Latent Semantic Indexing (LSI) method assumes that there is an underlying latent semantic space

that it estimates by means of statistical techniques using an association matrix ($n \times m$) of terms-documents: Documents in our case correspond to WordNet senses. In our case the semantic space is constructed by terms in the *vicinity* of the senses $S_1, S_2, \dots, S_m$ of the WordNet entry matching an ontology class name C , or a WSDL message parameter name C . In case a name is a compound word, then C is the right-most word in this compound. The set of terms in the semantic space include:

- The term C' that corresponds to C . C' is a lexical entry in WordNet that is a typographic variation of C .

- Terms that appear in C' WordNet senses $S_1, S_2, \dots, S_m$.

- Terms that constitute hyperonyms / hyponyms of each C' sense.

- Terms that appear in hyper(hyp)onyms of C' senses.

Latent Semantic Analysis (LSA) computes the arrangement of a k -dimensional semantic space to reflect the major associative patterns in the data. This is done by deriving a set of k uncorrelated indexing factors (latent features).

When one searches an LSI-indexed database of documents, it provides a query (i.e. a pseudo-document), which is a list of terms. In our case, each query is being constructed by extracting terms from the label, and the comment of an ontology class, as well as from the names of its properties'. Concerning the WSDL specifications, for each message parameter we extract terms from its "name" attribute, as well as from its related textual annotations. The similarity between two documents is computed by means of the dot product between the corresponding representation vectors. Doing so, LSI returns a set of graded documents, according to their similarity to the query. By virtue of dimension reduction from the N terms space to the k factors space, where $k < N$, terms that did not actually appear in a document may still end up close to the document, if this is consistent with the major patterns of association in the data.

Finally, we have also used a vector space model-based method (VSMM), which computes the matching of documents' pairs. Each document is represented by a vector of n weighted index terms. Index terms correspond to the simple terms that are extracted from all documents. Here we construct (pseudo-)documents that correspond to ontology classes and WSDL messages' parameters: As it is done in the LSA-based method, these pseudo-documents include terms from the label, and the comment of an ontology class, as well as from the names of its properties'. Moreover, the VSMM method enriches the pseudo-document with terms derived from the super-classes of a class. Concerning the WSDL specifications, for each input/output parameter we extract terms from its name, from the names of the elements that are the -simple or complex- types of the parameter, as well from the textual annotations associated with each element. The VSMM builds the vector of a pseudo-document by assigning to the weight of a term the frequency of its appearance in the pseudo-document. The similarity between two vectors (each corresponding to a WSDL message parameter and to an ontology class) is computed by means of the cosine similarity measure. We need to point out that VSMM aims to address problems similar to the ones that LSA-based-mapping aims to tackle, but exploiting different facets of information available in the WSDL specification and the ontology.

4.3 Combination of the Mapping Methods

During the first execution of the overall method there are no known cases (i.e. training examples for the learning method). In this case, the individual mapping methods are combined by taking the geometric mean of the "scores" calculated for each potential mapping pair. For each WSDL parameter this is done as follows:

1. For each mapping method the scores of the potential mapping pairs (WSDL parameter-name/OWL class) are computed.
2. The scores are normalized in the range $[0,1]$ by dividing each score with the highest one.

3. The Laplace smoothing law by means of the following function is applied in order to eliminate zero score values.

$$C_{smooth,i} = ((C_i / C_{max}) + 1) / n + 1 \quad (1)$$

C_i is the score computed by the i -th method, “ n ” is the number of scores to be smoothed.

4. The similarity scores produced by each method - for a specific potential mapping pair - are multiplied and raised to the n th root, where n is the number of distinct methods being combined.

$$C_{GM} = \left(\prod_{i=1}^n C_{smooth,i} \right)^{1/n}$$

Finally, the pairs with the k highest scores are selected.

The geometric mean (GM) provides a way of combining different methods, assuming that all methods contribute equally to the final mapping decision: This simple method is not accurate enough to automate the semantic annotation process or assist human annotators. This is evidenced by the results provided in the experimental evaluation section. To increase the accuracy of the overall method and its potential to exploit the results of the individual mapping methods in a “better” way, subsequent executions of the overall method use a gradient boosting method (GB) for learning how to combine the individual mapping methods.

In more detail, gradient boosting constructs additive regression models by sequentially fitting a simple base learner to current pseudo-residuals by least squares at each iteration. The pseudo-residuals are the gradient of the loss function being minimized, with respect to the model values at each training data point, evaluated at the current step. Specifically, the algorithm, originally proposed in [11], is as follows:

1. $F_0(\bar{x}) = \arg \min_{\gamma} \sum_{i=1}^N \Psi(y_i, \gamma)$
2. For $m=1$ to M do:
3. $\tilde{y}_{im} = - \left[\frac{\partial \Psi(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x_i)=F_{m-1}(\bar{x})}, i = 1 \dots N$
4. $\{R_{lm}\}_1^L = L - \text{terminal} - \text{node} - \text{tree}(\{\tilde{y}_{im}, \bar{x}_i\}_1^N)$
5. $\gamma_{lm} = \arg \min_{\gamma} \sum_{x_i \in R_{lm}} \Psi(y_i, F_{m-1}(\bar{x}_i) + \gamma)$
6. $F_m(\bar{x}) = F_{m-1}(\bar{x}) + \nu \cdot \gamma_{lm} \cdot 1(\bar{x} \in R_{lm})$
7. end for

where, F_i $i=1..M$, is the estimated function (mapping $\bar{x}$ to y values) during the i -th iteration (F_0 is an initial guess), Ψ is the loss function whose expected value must be minimized, ν is a shrinkage parameter in (0,1) that controls the learning rate of the algorithm, R_{lm} is the l -th partition of an L -terminal node regression tree during the m -th iteration of the algorithm, and $\{y_i, \bar{x}_i\}_1^N$ are N known $\{y, \bar{x}\}$ -values.

In our case, the known values (training examples) are of the form $\{y, \bar{x}\} = (y, (C_{COCLU}, C_{LEVEN}, C_{LSA}, C_{VSMM}, C_{GM}))$, where C_{METHOD} is the assessment (score) of each individual method for a specific mapping pair (a value in [0,1]) and y is in $\{0,1\}$, depending on the output of the human validation stage (1 if the mapping is a correct one, 0 otherwise). Given these known tuples, the gradient boosting method aims to construct additive regression models (line 6 of the algorithm) by fitting a simple parameterized function (lines 4 and 5 of the algorithm) to current pseudo-residuals (line 3 of the algorithm). Actually, in line 4 the method constructs a regression tree that models a base learner, given the training examples.

In our experiments we have used the implementation of the additive regression method provided in WEKA with shrinkage parameter equal to 0.1.

5. EMPIRICAL STUDY

5.1 Corpus Characteristics and Variations

To evaluate the proposed approach, we have used the SAWSDL-TC test collection [5]. A representative subset of this collection (244 services organized in 7 domains) was selected and provided for experimentation purposes by the SMR2 Workshop and S3 (<http://projects.semwebcentral.org/projects/owl-s-tc/>) contest organizers.

WSDL messages’ parameters as well as messages, operations and services have been manually annotated by a person who has been advised to carefully choose the annotations in order to indicate as close as possible the intended meaning of the annotated WSDL specifications. Where possible, the annotator advised to get consultation from xsd-schema complex types included in the WSDL specifications, or from external sources, such as lexicons. The annotator followed a distinct strategy: He provided commonsense definitions, “studying” the context of specifications (i.e. the complex types, the corresponding domain ontologies and the service as a whole), trying to “simulate” the textual annotation that would be done by the service developer/provider.

Table 1 presents information concerning the services and the ontologies used in our experiments. Concerning the services’ specification documents related to domains 1, 2, 3 and 6, the messages’ parameter names are mainly composed by a single-word term capitalized and an underscore character as a prefix (e.g. _COUNTRY). For the domains 4, 5 and 7, the messages’ parameter names are composed by multi-word terms (e.g. GetPatientMedicalRecords_AuthorizedMedicalRecords).

It must be noticed that since names of WSDL elements are very informative, and most of the parameter names in the original corpus differ from ontology class names by an underscore (“_”), we provide experimental results for different variations of the corpus.

The Lex variations contain lexical/typographic alterations (e.g. PRC instead of PRICE) for a percentage of the parameter names. Also, in the Syn variations of the corpus a percentage of parameter names (e.g. HOTEL) have been replaced by a synonym term, or by a compound term (e.g. GuestHouse) whose meaning is close to the meaning of the original name. Finally, in the Mix variations, a percentage of parameter names have been replaced with synonyms and these have been changed to a typographic variation. For each of the above types of variations we provide two corpora with changes in 30% and 60% of the parameter names. Thus, distinguishing the seven domains (the method runs for each of the domains separately) we have in total 42 (7*3*2) domain-specific corpora.

Table 1. Quantitative information about the corpus used in the experiments.

Domain	Services	Domain Ontology (.owl)		
		#ontology	#concepts	#properties
(1) Communication	13	5	366	32
(2) Economy	65	10	1329	366
(3) Education	65	7	1150	346
(4) Food	15	4	822	211
(5) Medical	37	11	1172	425
(6) Travel	37	5	835	323
(7) Weapon	12	3	655	225

Generally, ontologies accompanying the service collection have been developed independently from the corpus. Specifically, there are ontologies with some annotations for their classes (e.g. the

Travel ontology provides comments for the classes, and the Portal ontology provides comments or labels for some of the classes), and ontologies with no annotations at all (e.g. the HospitalPhysician). Also, the elements of some ontologies are highly interrelated, such as in SUMO (max parents: 3, mean parents: 2, max siblings: 15, mean siblings: 7, all properties have a domain and range specified), but this is not true for others, such as for the Books ontology (there is no property with specified domain and range).

To simulate the iterative execution of the overall method and study its gradual improvement via exploiting known mappings as training examples, we performed experiments in which: (a) 25% (b) 50%, and (c) 75% of the gold mappings are known per domain. The mappings known in cases (a) and (b) are chosen randomly and are included in those known in cases (b) and (c), respectively. This simulates the gradual execution of the method for a set of services in a specific time period. To further increase our confidence to the results, we performed 5 different experiments with randomly chosen sets of known mappings for each of these cases. Having said that, we have to point out that as shown in Table 1, the number of services, and thus parameters, in some of the domains is very small.

The training examples (known cases) are generated as follows: Positive examples are the gold mappings (i.e. have been provided as correct mappings to the method). In this case $y=1$. Additionally, for every positive example, the highest ranked wrong mapping produced by GM is added to the training dataset (y is set to 0), as well.

Furthermore, GM has been used as a baseline method to show the benefits of the proposed method. We measure the accuracy of each method (GB and GM) in terms of macro-recall, in every variation of the corpus, for every domain. The macro-recall is defined as the average of the recall measure produced for each WSDL document. Recall is defined as the ratio of correctly produced mappings to all the correct mappings. The gold mapping pairs were derived from the exploitation of the SAWSDL:modelReference attribute in services' specifications.

Furthermore, for each algorithm we provide top-k macro-Recall for every corpus variation, where k is equal to 3, 5 and 10.

5.2 Results and Discussion

The hypotheses that we need to test during experimental evaluation are as follows:

(a) The gradient boosting algorithm manages to gradually improve the accuracy of the mapping mechanism as new mappings are known. This happens in all variations of the corpus.

(b) The mapping mechanism enhanced with the gradient boosting algorithm achieves higher accuracy than when the scores of methods are combined using the geometric mean.

(c) The gradient boosting algorithm manages to improve the accuracy of the algorithm even when the number of parameters (and thus, training examples) is low. In domains with a larger number of parameters, this happens in a higher "degree"

In the diagrams that follow, the x-axis shows the percentage of the known values, and the y-axis shows the macro-recall achieved. Notice that the minimum and higher values in the y-axis vary per diagram. In each diagram there are results from different variations of the corpus, denoted by "X Y - Z k", where X is in {Lex, Syn, Mix} and Y is in {30%, 60%}. Z shows how the compositions of the individual mappings have been calculated: Either by gradient boosting (GB) or by geometric mean (GM). The "k" shows the number of highly ranked results returned, and thus it is in {3, 5, 10}.

Figure 2 shows the average macro-recall of the method for all domains. Specifically, in every variation of the corpus the mapping algorithm with the gradient boosting (GB) method manages to improve its accuracy as new mappings are known and used as training examples. This is in contrast to the combination of

mapping methods via the geometric mean (GM) method, whose accuracy is lower than that in the corresponding GB experimental cases (except in very few cases), and it does not improve. Actually, only in 14 out of 42 cases the macro-recall of GM is greater than the accuracy of the GB: The average difference for these cases is 0.017, and the higher one is 0.050.

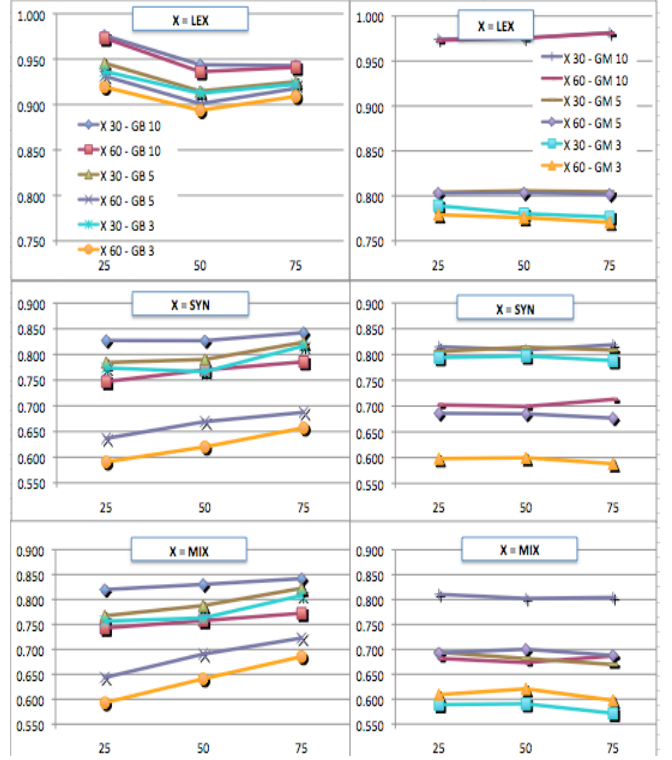


Figure 2. Overall performance of the GB mapping method (left column) against the GM combination of mapping methods (right column)

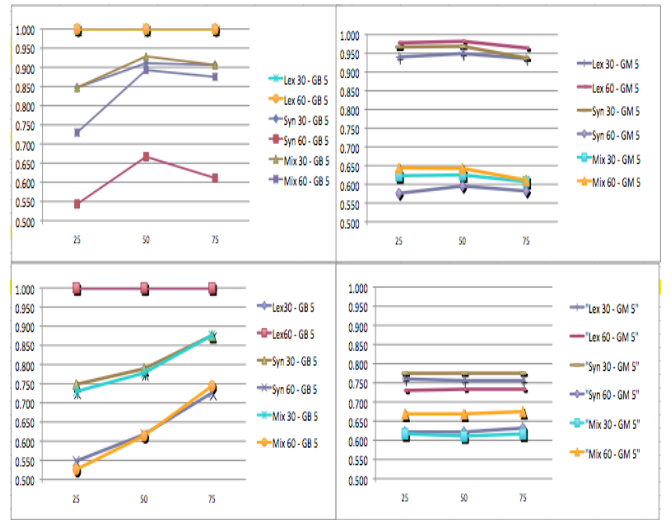


Figure 3. Performance of the GB mapping method (left column) against the GM combination for two domains: Communication (1st row) and Education (2nd row)

The small decrease of accuracy for the GB method in Lex variations from 25% to 50% (less than 0.037) occurs since correct mappings in these cases have been ranked in lower positions in the list of ranked mappings. However, the top-3 and top-5 accuracy of GB is better than GM in all cases: This provides clear evidence to

the superiority of this method. Furthermore, the macro-recall of the mapping methods achieved in the Lex variations of the corpus is slightly higher than that in the Syn variation, which is slightly higher than that in the Mix variations: This holds since the difficulty of computing mappings, from Lex to Syn and from Syn to Mix increases. This shows that difficulties concerning the computation of correct mappings are addressed successfully.

It must be noticed that the rate at which the macro-recall increases for GB as the percentage of known cases increases, is nearly the same for all corpus variations and it is not very high. This rate is different for each of the domains with different number of services (and thus, parameters).

Figure 3 shows the results from two specific domains in the corpus: The “Communication” and the “Education”. Experimental results concern all variations of the corpus, for $k=5$. We show the results for $k=5$ as it is not that strict as $k=3$, and not that tolerant as $k=10$. The first domain has very few services and thus, the potential of the GB to increase the mapping macro-recall is expected to be low. On the other hand, the “Education” domain is one of the largest in the corpus and thus, GB is expected to increase its macro-recall of the mapping method as new cases are known.

According to Figure 3 there is a slight decrease (less than 0.05) of the accuracy of the GB method when the known mappings increase from 50% to 75% in the Communication domain. Also, it is evident that GM performs slightly better than GB in the Syn60 variation of the corpus when 75% of the mappings in the Communication domain are known. In the Education domain, when 25% of the mappings are known, GM achieves higher accuracy for the Syn30 and Syn60 variations of the corpus. However GM in the next executions of the method it is outperformed by GB.

Finally, the pair-wise two-tailed t -test results under significance level of 0.05 are summarized in Table 2, where ‘win/loss’ means that the gradient boosting (GB) algorithm is significantly better/worst, respectively, than the geometric mean (GM) method, and ‘tie’ means that there is no significant difference between GB and GM. Table 2 shows that GB is better than GM as it manages to perform 9 significant wins and 2 significant losses in the 25% corpus variation.

Table 2. Significant wins/ties/loss of GB against GM.

Percentage of initial number of parameters	win/tie/loss
25%	9/7/2
50%	12/2/4
75%	16/0/2

This ratio increases as the number of known correct cases increases, while the number of ties decreases, making clearer the superiority of the gradient boosting algorithm. Overall, Table 2 shows that the GB method increases the effectiveness of the mapping task. It also shows that the accuracy increases as the number of known cases increases.

6. CONCLUSIONS

Efficient and effective semantic annotation of services can be done semi-automatically in cases where (a) methods automatically map services’ elements to ontology elements with high accuracy, (b) humans validate the mapping results and (c) methods learn from the validated mappings so as to increase their effectiveness. In this paper we have demonstrated how this can be done for “uncovering” the data semantics of WSDL specifications by combining a set of methods that map messages’ parameters to ontology concepts and by learning how to combine these methods so as to increase the effectiveness of the overall semantic annotation task.

Further work concerns increasing the efficacy of the learning mechanism further, choosing the “right” ontologies for mapping WSDL elements and implementing a toolkit where one may choose the combination of different mapping methods so as to produce semantic annotations.

REFERENCES

- [1] A. Patil, S.Oundhakar, A.Sheth and K.Verma, METEOR-S Web Service Annotation Framework, WWW 2004, pp. 553-562, (2004).
- [2] E. Stroulia, Y. Wang, Structural and Semantic Matching for Assessing Web-Service similarity, IJCIS 5:14, pp. 1-30 (2004).
- [3] M. Klusch, B. Fries, M. Khalid and K. Sycara OWLS-MX: Hybrid OWL-S Service Matchmaking, AAAI (2005).
- [4] M. Klusch and B. Fries, Hybrid OWL-S service retrieval with OWLS-MX: Benefits and Pitfalls, SMR2, (2007).
- [5] SAWSDL-TC, <http://projects.semwebcentral.org/projects/sawSDL-tc/>
- [6] A.Valarakos, G. Paliouras, V. Karkaletsis and G. Vouros, A Name-Matching Algorithm for Supporting Ontology Enrichment, SETN (2004)
- [7] A. Heß, E. Johnston and N. Kushmerick, ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services, ISWC (2004).
- [8] U. Keller, R. Lara, H. Lausen, A. Polleres and D. Fensel, Automatic Location of Services, ESWC’05, (2005)
- [9] C. Ringelstein, T. Franz, and S. Staab, The Process of Semantic Annotation of Web Services, Semantic Web Services - Theory, Tools, and Applications, Idea Publishing Group, USA, (2007)
- [10] K. Kotis, G. Vouros and K. Stergiou, Towards Automatic Merging of Domain Ontologies: The HCONE-merge approach, Elsevier's Journal of Web Semantics (JWS), vol. 4:1, pp. 60-79 (2006)
- [11] Friedman J.H. Stochastic Gradient Boosting. Tech. Report Stanford University. <http://www-stat.stanford.edu/~jhf/ftp/stobst.ps> (1999)